View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2007 Proceedings

European Conference on Information Systems (ECIS)

2007

A Truthful Heuristic for Efficient Scheduling in Network-Centric Grid OS

Jochen Stoesser
Information Management and Systems, University of Karlsruhe, stoesser@iism.uni-karlsruhe.de

Dirk Neumann
Institute of Information Systems and Managemen, neumann@iism.unikarlsruhe.de

Arun Anandasivam
Institute of Information Systems and Managemen, anandasivam@iism.unikarlsruhe.de

Follow this and additional works at: http://aisel.aisnet.org/ecis2007

Recommended Citation
Stoesser, Jochen; Neumann, Dirk; and Anandasivam, Arun, "A Truthful Heuristic for Efficient Scheduling in Network-Centric Grid OS" (2007). *ECIS 2007 Proceedings*. 176.
http://aisel.aisnet.org/ecis2007/176

This material is brought to you by the European Conference on Information Systems (ECIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

A TRUTHFUL HEURISTIC FOR EFFICIENT SCHEDULING IN NETWORK-CENTRIC GRID OS

Jochen Stoesser, Institute of Information Systems and Management (IISM), Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany, stoesser@iism.uni-karlsruhe.de

Dirk Neumann, Institute of Information Systems and Management (IISM), Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany, neumann@iism.uni-karlsruhe.de

Arun Anandasivam, Institute of Information Systems and Management (IISM), Universität Karlsruhe (TH), Englerstr. 14, 76131 Karlsruhe, Germany, anandasivam@iism.uni-karlsruhe.de

Abstract

The Grid is a promising concept to solve the dilemma of increasingly complex and demanding applications being confronted with the need for a more efficient and flexible use of existing resources. Network-centric Grid Operating Systems (OS) aim at providing users and applications with transparent and seamless access to heterogeneous Grid resources across different administrative domains. Scheduling in these heterogeneous environments becomes the key issue since scarce resources must be distributed between strategic and self-interested users. Market-based algorithms are deemed to provide a good fit to the Grid environment by accounting for its distinct properties and the needs of the users. Current market mechanisms, however, leave room for inefficiency and are computationally intractable. The speed of heuristics becomes a desirable feature in interactive, large-scale Grid OS settings. Heuristics achieve a fast, approximately efficient allocation of resources but generally fail in preserving truthfulness, i.e. users can benefit from cheating the mechanism. The contribution of this paper is the proposal of a greedy, market-based scheduling heuristic for network-centric Grid OS which does achieve this distinct trade-off: it is designed so as to obtain an approximately efficient allocation schedule at low computational cost while accounting for strategic, self-interested users in a heterogeneous environment.

Keywords: network-centric Grid OS, market-based scheduling, heuristic, truthful

1 INTRODUCTION

Grids allow the aggregation and sharing of a wide variety of geographically dispersed computer resources owned by different administrative units (Foster and Kesselman 2004). Grids are typically used for scientific applications that require massive amounts of computational power and storage such as protein folding, weather forecasts, or gene encoding. These applications potentially process and store terabytes of data and the sharing of resources is necessary to reduce the enormous processing time (Berlich et al. 2005; Cirne et al. 2006). The most famous predecessor of modern Grids is SETI@home which connected in its peak time over one million computers spread across 226 countries to achieve processing power of 418.6 TFLOPS (Cirne et al. 2006). For comparison, the world's fastest supercomputer – IBM's BlueGene/L system – has an estimated total processing power of 280 TFLOPS, while Google's search engine system can muster between 126 and 316 TFLOPS. Grids offer enormous potential by aggregating idle resources that are geographically distributed. For businesses, it is projected that Grids may reduce total IT costs by 30 % (Minoli 2004).

The use of market mechanisms has often been suggested in situations that establish a price for resources. The price being paid to resource owners that share idle resources reflects the scarcity of the provided resource. Since consuming resources is not for free, demand will be restricted and shifted to those time slots where the resources are really needed. The employment of market mechanisms in distributed computing is not new. The first attempt was reported at Harvard University where auctions were used to allocate computing time to users of the PDP-1 computer (Sutherland 1968). Other examples are Tycoon, Bellagio, and Nimrod/G (Buyya et al. 2000; Lai et al. 2004).

From an economic point of view, the use of market mechanisms in Grid computing is promising, but most of the potential in state-of-the-art Grid middleware such as Globus, UNICORE, and gLite is being wasted. Current Grid middleware is based on the premise to share *idle* resources only and computational jobs are submitted to these idle resources where they are being processed in batch mode. It would be more efficient to allocate all available resources over markets. Ideally, resource allocation should be on-demand such that interactive processing of jobs is permitted as well (Gorlatch and Müller 2005).

Recently, a new stream of research on network-centric Grid operating systems (OS) has emerged in Grid computing that especially addresses interactive applications. Network-centric Grid OS provide the user with seamless and transparent access to Grid resources such as memory and computing power. While current OS provide only limited support for Grid computing, network-centric Grid OS adjust their view on the system at run-time contingent upon the particular resources. By means of network-centric Grid OS, the Grid can be used in an interactive manner. This interactivity certainly raises problems concerning bandwidth and latency – network-centric-Grid OS are an active research topic (Padala and Wilson 2003; Gorlatch and Müller 2005). Nonetheless, there are currently some implementations available that come close to this notion of a network-centric Grid OS. MOSIX, for example, is realized as an OS virtualization layer that provides a single system image with the Linux run time environment to users and applications. As such, it allows applications to run on remote nodes as if they ran locally. Users can run their applications while MOSIX is automatically and transparently seeking resources and migrating processes among nodes in order to improve overall performance (Barak et al. 2005).

The concept of network-centric Grid OS coupled with markets is deemed promising to revolutionize resource allocation in Grids. From their conception, network-centric Grid OS do not only allocate resources being idle but all available resources. The employment of markets for allocating and scheduling jobs to resources appears promising to achieve an efficient resource allocation in the economic sense – allocating those jobs to the resources that value it most.

The problem of scheduling resources is, however, computationally demanding. To be harnessed for network-centric Grid OS, market mechanisms needs to be solvable quickly. Currently, there is no market mechanism available that is specifically tailored towards the needs of a network-centric Grid

OS. This paper seeks to remedy this gap by designing a truthful scheduling heuristic for MOSIX that achieves an approximately efficient allocation fast. The contributions of this paper are threefold: Firstly, the paper designs a multi-attribute exchange that can be used for Grid OS. Secondly, a greedy heuristic is employed to solve the scheduling problem. Thirdly, an adequate pricing scheme is developed which assures that reporting truthfully is a dominant strategy for resource requesters and payments to resource providers are approximately truthful.

The remainder of this paper is structured as follows. In Section 2, the market-based scheduling problem of network-centric Grid OS systems is introduced. Section 3 suggests (i) a heuristic to solve this problem and (ii) a pricing scheme that provides the market mechanism with desirable properties. Section 4 discusses related work in the light of the presented market mechanisms. Section 5 concludes the paper with a summary and points to future work.

2 MARKET-BASED SCHEDULING IN GRID OS

There are two classes of system users in network-centric Grid OS: Resource requesters who want to use the computing resources offered in the system for solving a computational problem and resource providers who want to sell idle resources. We introduce the term "job" to refer to a computational problem and we will call an atomic bundle of resources on which the job or parts of it can be computed a "node", e.g. one server within a cluster. If a job gets allocated to a node, we will call this job (node) a "winning" job (node). We intend to design a direct, sealed-bid mechanism which allocates periodically: the mechanism collects resource requests and offers from the users for a period of time and then allocates jobs to nodes on the basis of these submitted requests and offers. Users do not get to know the requests and offers submitted by other users before the allocation is determined by the mechanism.

2.1 Economic Requirements

The mechanism is intended to perform job scheduling in a distributed computing environment with heterogeneous and selfish users. There are a number of economic requirements which a market-based scheduling mechanism should ideally satisfy in this environment (Schnizler et al. forthcoming):

- **Allocative efficiency:** A mechanism is said to be allocatively efficient if it maximizes the utility across all participating users (welfare), i.e. the sum over the valuations of all winning resource requesters less the sum over the reservation prices of all winning resource providers.
- **Budget-balance:** the mechanism does not need to be subsidized by outside payments. The payments coming from the resource requesters cover the payments made to the resource providers.
- **Computational tractability:** the mechanism can be computed in polynomial run time in the size of its input, i.e. the number of resource requests and offers.
- **Truthfulness:** it is a (weakly) dominant strategy for users to reveal their true valuations to the mechanism.
- **Individual rationality:** users cannot suffer any loss in utility from participating in the mechanism, i.e. it is individually rational to participate.

Budget-balance and individual rationality are hard constraints which the mechanism must suffice. If these two requirements are not met, the mechanism will not be sustainable. Truthfulness is a desirable feature since it tremendously simplifies the strategy space of the users; there is no need to reason about the strategies of other users. Due to the celebrated impossibility result of Myerson and Satterthwaite (1983), this inherently implies that allocative efficiency can only be approximated: There is no exchange which is at the same time budget-balanced, individually rational, truthful and allocatively efficient in equilibrium.

There are three basic components to be designed: the *bidding language* which defines how requests and offers are specified, the *allocation algorithm* which decides which job is to be executed on which node at what time, and the *pricing scheme* which translates the resulting allocation schedule in corresponding monetary transfers. In this section, the bidding language and the winner determination problem will be formalized. The complexity of this scheduling problem forms the rationale for a market-based heuristic. The pricing scheme will only be specified for the heuristic due to the focus of this paper.

2.2 Bidding Language

There is on-going research on applying machine learning, regression and filtering techniques for predicting run times of future jobs (Ali et al. 2004). In many cases a certain resource requester will submit jobs to a distributed computing environment which are similar in terms of algorithms, data structures, job size etc. Thus the run time of a new job is likely to follow the run times of "similar" jobs in the past. Building on this research we will assume that the resources needed for the computation of a job are known to the requester a priori. This comprises the amount of computing power, memory, and the run time of the job. A resource requester wanting to compute a job j submits a request $(b_j, \underline{c}_j, \underline{m}_j, s_j, e_j)$ to the market mechanism where $b_j \in \Re^+$ denotes the requester's willingness to pay per unit of computing power and time slot, $\underline{c}_j \in \aleph$ and $\underline{m}_j \in \aleph$ the minimum amount of computing power and memory, $s_j \in \aleph$ the time slot at which the job needs to be started, and $e_j \in \aleph$ the time slot until which the job needs to be run. A job can only be executed in its entireness (atomicity). It cannot be parallelized on multiple nodes but migrate across different nodes over time. Note that *job migration* is one of the main features which distinguish Grid OS in general and MOSIX in particular from traditional machine scheduling domains. A resource offer containing node n consists of a tuple $(r_n, \overline{c}_n, \overline{m}_n, \varepsilon_n, \lambda_n)$ where $r_n \in \Re^+$ denotes the reservation price per unit of computing power and time slot, $\overline{c}_n \in \aleph$ and $\overline{m}_n \in \aleph$ the maximum amount of computing power and memory, and $\varepsilon_n \in \aleph$ and $\lambda_n \in \aleph$ the earliest and the latest time slot at which the resources are available. Contrary to atomic resource requests, resources offered by some node are divisible and moreover freely disposable.

Example: The following resource requests and offers have been submitted to the system:

Job j	b_j	$\underline{c}_j$	$\underline{m}_j$	s_j	e_j	Node n	r_n	$\overline{c}_n$	$\overline{m}_n$	ε_n	λ_n
J1	11	60	50	2	3	N1	5	150	100	1	5
J2	10	80	50	1	3	N2	7	100	120	2	6
J3	9	90	100	2	5	N3	8	140	100	1	5
J4	8	130	100	4	5						
J5	7	80	90	1	3						
J6	6	70	50	2	4						
J7	5	30	40	1	3						

Table 1. Sample resource requests and offers.

Job J1 requests to be run in time slots 2 and 3 and requires at least 60 units of computing power and 50 units of memory in each time slot. The resource requester for job J1 is willing to pay up to \$11 per unit of computing power and time slot, i.e. \$11 * 60 * 2 = \$1,320 in total. The provider of node N1 offers at most 150 units of computing power and 100 units of memory in time slots 1 to 5 and requires a minimum payment of at least \$5 per unit of computing power per time slot.

To simplify notation, in the following we will assume that each request and each offer has been submitted by a separate user and we will use the terms request and job (offer and node) interchangeably. In contrast to Schnizler et al. (forthcoming), this bidding language does not support a combinatorial exchange where users can request and offer arbitrary sets of goods and possibly link multiple requests and offers by means of logical operators. In the scenario at hand, each request and offer specifies one and the same bundle of computing resources and memory and thus rather implements a multi-attribute exchange (Bichler et al. 1999) by allowing the specification of multi-unit bundles, e.g. 100 units of computing power and 50 units of memory. Note, however, that while the bidding language is not combinatorial, the scheduling problem of allocating jobs to nodes remains a combinatorial assignment problem as will be seen below.

2.3 Exact Scheduling

Exact mechanisms such as the well-known Vickrey-Clarke-Groves (VCG) mechanism are guaranteed to find an optimal solution to the scheduling problem. Let N be the set of nodes contained in resource offerings, J the set of jobs contained in resource requests, $T^S(n) := \{t \in \aleph \mid \varepsilon_n \leq t \leq \lambda_n\}$ the set of time slots in which node $n \in N$ is available according to the respective resource offering, $T^b(j) := \{t \in \aleph \mid s_j \leq t \leq e_j\}$ the set of time slots in which job $j \in J$ requests to get executed, and $N(j) := \{n \in N \mid T^b(j) \subseteq T^S(n), \underline{c}_j \leq \overline{c}_n, \underline{m}_j \leq \overline{m}_n\}$ the set of nodes which can potentially execute job j. We introduce the binary decision variable $x_{jnt} \in \{0,1\}$ with $x_{jnt} = 1$ if job j will be executed on node n in time slot t and $x_{jnt} = 0$ otherwise. The scheduling problem can be formalized as the following mathematical integer programme:

$$\max_{x_{jnt}} V := \sum_{j \in J} \underline{c}_j \sum_{t \in T^b(j)} \sum_{n \in N(j)} x_{jnt}(b_j - r_n)$$

[SP]

$$s.t. \quad x_{jnt} \in \{0,1\}, j \in J, t \in T^b(j), n \in N(j) \qquad (C1)$$

$$\sum_{n \in N(j)} x_{jnt} \leq 1, j \in J, t \in T^b(j) \qquad (C2)$$

$$\sum_{j \in J} x_{jnt} \underline{m}_j \leq \overline{m}_n, n \in N, t \in T^S(n) \qquad (C3)$$

$$\sum_{j \in J} x_{jnt} \underline{c}_j \leq \overline{c}_n, n \in N, t \in T^S(n) \qquad (C4)$$

$$\sum_{u \in T^b(j)} \sum_{n \in N(j)} x_{jnu} = (e_j - s_j + 1) \sum_{n \in N(j)} x_{jnt}, j \in J, t \in T^b(j) \qquad (C5)$$

$$b_j \geq x_{jnt} r_n, j \in J, n \in N(j), t \in T^b(j) \qquad (C6)$$

SP assigns jobs to nodes as to maximize welfare V. The economic scheduling scenario is encoded in the constraints of this combinatorial assignment problem as follows: (C1) introduces x_{jnt} as binary decision variable. Jobs can only be assigned to nodes which provide the necessary resources during the right time slots. (C2) ensures that a job is allocated to at most one node at a time. (C3) and (C4) specify that for any given time slot and node, the total resource consumption in terms of memory and computing power cannot exceed the resources provided by this node. (C5) defines atomicity, i.e. every job is either executed as a whole or it is not executed at all. (C6) ensures that for each allocation of a job to a node, the requester's willingness to pay is equal to or greater than the provider's reservation price. The optimal allocation schedule $X^* := (x^*)_{j \in J, n \in N, t \in Tb(j)}$ yielding welfare V^* can be derived directly from the solution to SP.

Example: For the sample resource requests and offers listed in Table 1, building and solving SP yields the optimal allocation schedule X^* shown in Figure 1. In time slot 1, only job J2 runs on node N1. The requester of job J2 is willing to pay up to \$10 [per computing cycle] * 80 [computing cycles] = \$800 per time slot. The provider of node N1 requires a minimum payment of \$5 per computing cycle and time slot and receives – depending on the pricing which will be elaborated below – at least \$5 * 80 = \$400. Thus the allocation of job J2 to node N1 in time slot 1 yields welfare of \$800 - \$400 = \$400. Across all time slots, X^* generates welfare of V^* = \$3,420.

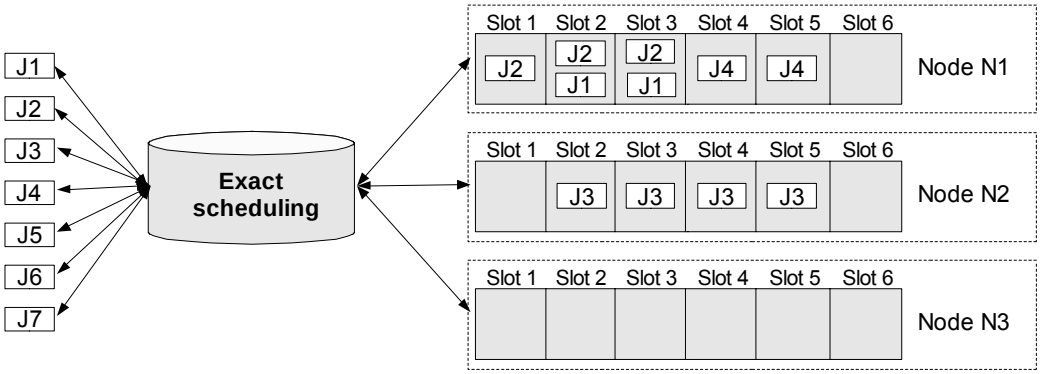

Figure 1. Optimal Allocation Schedule.

While this exact mechanism generates an optimal allocation schedule in terms of economic efficiency, this optimality only comes at the expense of computational efficiency: SP is an instance of a multiple knapsack problem (Ferrari 1994) and thus NP-hard. Computational costs of determining the allocation and the pricing may become a problem in dynamic network-centric Grid OS with a large number of resource requests and offers (Waldspurger et al. 1992). In these settings, a trade-off between efficiency and computational complexity may be desirable to support interactive applications.

3 TRUTHFUL SCHEDULING HEURISTICS

Heuristics aim at obtaining a good but generally suboptimal (in terms of welfare) allocation fast. Relaxing allocative efficiency not only impacts the outcome determination but also the pricing scheme: compromising a mechanism in terms of allocative efficiency generally implies compromising truthfulness as well (Mu'alem and Nisan 2002). Lehmann et al. (2002) and Mu'alem and Nisan (2002), however, elaborate necessary and sufficient conditions that do yield truthful, approximating mechanisms for a restricted class of users: *known single-minded bidders*. Informally, a bidder (user) is known single-minded if she only desires one specific set of goods and this set is known to the mechanism. Applying these conditions to the scenario above in order to design a truthful mechanism suggests itself: in a non-combinatorial multi-attribute exchange such as a market-based scheduler for Grid OS, resource requesters and providers are only allowed to request and offer one single bundle of resources corresponding to the bundle of attributes – in the scenario at hand this is computing power and memory.

3.1 A Greedy Scheduling Heuristic

Lehmann et al. (2002) and Mu'alem and Nisan (2002) propose greedy heuristics for constructing truthful heuristics for single-unit combinatorial auctions with one seller and multiple buyers. We build on this proposal and apply this greedy principle to construct a heuristic for the multi-attribute Grid exchange at hand with multiple sellers and multiple buyers. This heuristic consists of two basic phases:

1. Sort the requests $j \in J$ and offers $n \in N$ in non-increasing and non-decreasing order with respect to some norm η;
2. Sequentially run through the resulting rankings and allocate the requests with the highest ranking to the offers with the highest ranking (cf. pseudo code in Figure 2).

(1) Sort jobs $j \in J$ in non-increasing order and nodes $n \in N$ in non-decreasing order of η.
(2) Run sequentially through the job ranking, starting with the highest ranked job. For each job j:
(3) Run sequentially through the node ranking, beginning with the highest ranked node, and check if j can be executed, i.e. whether conditions $(b_j \geq r_m) \wedge (\underline{c}_j \leq \overline{c}_m) \wedge (\underline{m}_j \leq \overline{m}_m)$ are satisfied for some $m \in \{n_1,...,n_j\} \subseteq N$, i.e. the nodes with the highest ranking which can together accomodate job j, in time slots $t \in T^b(j)$.
(4) If so, allocate j to $\{n_1,...,n_j\}$; Update the residual capacities of $m \in \{n_1,...,n_j\}$, i.e. subtract $\underline{m}_j$ from $\overline{m}_m$ and $\underline{c}_j$ from $\overline{c}_m$ in time slots $t \in T^b(j)$.
(5) Continue at (2).

Figure 2. Heuristic for obtaining the initial allocation.

The allocative efficiency of this greedy heuristic essentially hinges on norm η used in the sorting phase (Lehmann et al. 2002). This norm must lead to "efficient" rankings in the sense that the heuristic can create an approximately efficient allocation based on these rankings. A straightforward choice is to use b_i and r_i respectively as these set the valuation for a job (node) in relation to the amount of resources requested (offered). In conjunction with the sequential allocation rule in phase two, the resulting heuristic truly implements a greedy allocation algorithm: in each allocation step, it intends to maximize $b_j - r_n$ from the objective function of SP above.

Example: Assume $\eta = b_j, j \in J$, and $\eta = r_n, n \in N$.

Job j	b_j	$\underline{c}_j$	m_j	s_j	e_j
J1	11	60	50	2	3
J2	10	80	50	1	3
J3	9	90	100	2	5
J4	8	130	100	4	5
J5	7	80	90	1	3
J6	6	70	50	2	4
J7	5	30	40	1	3

Node n	r_n	$\overline{c}_n$	$\overline{m}_n$	ε_n	λ_n
N1	5	150	100	1	5
N2	7	100	120	2	6
N3	8	140	100	1	5

Figure 3. Example of the greedy heuristic.

Jobs J1 and J2 can be fully executed on node N1. In time slots 2 and 3, job J3 does not fit on N1 but only on N2 which is next in the ranking. In time slots 4 and 5, J3 does fit on N1. Since J3 is allocated to N1 in time slots 4 and 5, J4 does not fit on N1 anymore but only on N3, in contrast to the optimal allocation X^*. In total, X^{greedy} yields welfare V^{greedy} = \$3,000 and an approximation ratio of V^{greedy} / V^* = \$3,000 / \$3,420 ≈ 88 %. The greedy heuristic generates the following allocation schedule X^{greedy} :

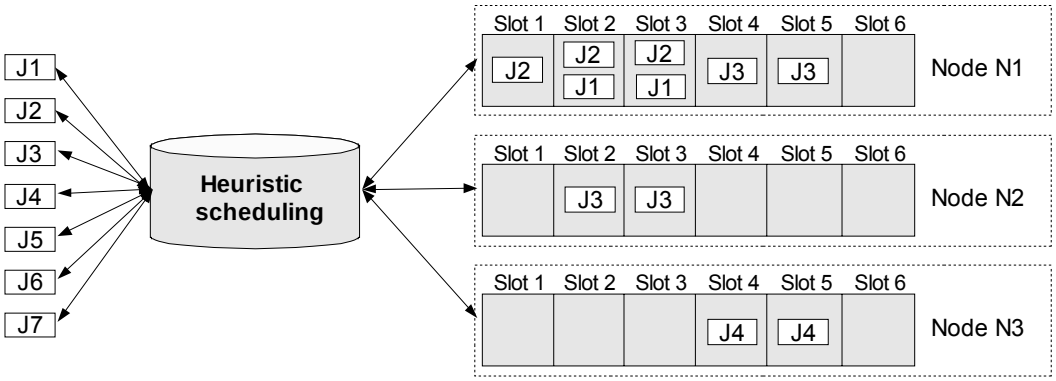

Figure 4. Greedy allocation schedule.

The question is: How can these allocations of requests to resources and vice versa be translated into corresponding monetary transfers (prices and payments) according to the desired design criteria introduced above? Mu'alem and Nisan (2002) and Lehmann et al. (2002) propose a pricing scheme which generates truthful prices in restricted one-sided, combinatorial auctions with one provider and multiple requesters. In the remainder, these theoretical results will be applied to the Grid OS context to generate truthful prices for resource requesters. In contrast to the setting of Mu'alem and Nisan (2002) and Lehmann et al. (2002), the Grid OS market consists of multiple requesters *and* multiple providers. An algorithm will hence be proposed which yields approximately truthful payments to resource providers.

3.2 Critical Value-based Pricing of Resource Requests

Following the spirit of the truthful Vickrey auction, the payment of job j corresponds to its *critical value* given sets J and N of jobs and nodes (Lehmann et al. 2002, Mu'alem and Nisan 2002). The critical value of ϕ_j is the minimal valuation that j has to report in order to remain in the allocation schedule, keeping all other requests and offers unchanged. Note that it is not possible to simply take the valuation of the job with the highest ranking which does not get executed since this job may not be executable in general due to capacity constraints. Moreover, removing j from the winning allocation might change the allocation of other jobs within the winning allocation as well. To determine the critical value for each winning job j, we need to determine the allocation *without* j: we successively allocate all other jobs from the ranking and, after having allocated a job, check whether j can still be accommodated (cf. pseudo code in Figure 5).

(1) Sort jobs $k \in J$, $k \neq j$, in non-increasing order and nodes $n \in N$ in non-decreasing order of η.
(2) Is any job $k \neq j$ left in the ranking?
If not, set $\phi_j = \max_{m \in \{n_1,\ldots,n_j\}} r_m$ where $\{n_1,\ldots,n_j\}$ are the nodes with the highest rankings which can together accommodate j, and finish.
If so, select job k with the highest ranking.
(3) Can k be accommodated? If not, continue at (2). If so, allocate k to the nodes $\{n_1,\ldots,n_k\} \subseteq N$ with the highest ranking that can accommodate k, and update the residual capacities of $m \in \{n_1,\ldots,n_k\}$.
(4) Can j still be accommodated on some nodes $\{n_1,\ldots,n_j\} \subseteq N$? If not, set $\phi_j = b_k$ and finish.
(5) If so, set $r := \max_{m \in \{n_1,\ldots,n_j\}} r_m$. Either $r < b_k$, then continue at (2); or $r \geq b_k$, then it is cheaper for j to push away k than to take the next available nodes $\{n_1,\ldots,n_j\}$; set $\phi_j = b_k$ and finish.

Figure 5. Determining the critical value of job j and refining the initial allocation.

The critical value must be computed once for each winning job j. However, compared to the computational intractability of the VCG mechanism, the computation now runs in polynomial time. Every winning job j has to pay $p_{greedy,j} = (e_j - s_j + 1)\underline{c}_j\phi_j$ whereas every rejected job pays nothing.

Example: Applying the algorithm to the example above, we get $p_{greedy,J1} = (e_{J1} - s_{J1} + 1)\underline{c}_{J1}\phi_{J1} = 120 \cdot b_{J7} = 120 \cdot r_{N1} = \$600$, i.e. the "cheapest" option for job J1 is to push away J7 by bidding \$5 (plus some ε) which happens to equal the minimum reservation price of node N1. In total, the mechanism collects a revenue R of \$6,400 from the resource requesters.

Job j	J1	J2	J3	J4
ϕ_j	5	5	7	8
$p_{greedy,j}$	600	1,200	2,520	2,080

Table 2. Payments of resource requesters.

The greedy allocation scheme combined with critical-value based pricing creates a truthful scheduling heuristic with respect to resource requesters: It is a weakly dominant strategy for resource requesters to report their true valuation for a job (the required resources). The payment depends on the critical value which is independent of the reported valuation. Assume a risk-neutral requester j has reported her true valuation v_j, i.e. $b_j = v_j$. Then there are four cases to be considered.

State\Action	Decrease b_j	Increase b_j
j was accepted	*Case 1*	*Case 2*
j was rejected	*Case 3*	*Case 4*

Table 3. Proof sketch.

Suppose j has won. Reporting a lower valuation (*Case 1*) might have resulted in j not being accepted while it would not have reduced j's payment. Reporting a higher valuation (*Case 2*) would not have generated more utility either since j has been accepted anyways.

Now suppose j has been rejected ($\phi_j > b_j = v_j$). Reporting a lower valuation (*Case 3*) would still have left j outside the winning allocation. Reporting a higher valuation (*Case 4*) might even have resulted in a loss: if j had been accepted, $b_j > \phi_j > v_j$.

3.3 Budget-balanced, Approximately Truthful Payments to Resource Providers

The concept of critical value-based pricing cannot be applied when determining the payments for divisible resource offers. The concept requires a binary decision in the sense that an offer is either fully executed or it is not executed at all. We allow divisibility of offers, i.e. not all offered resources need to be used but the offer can be executed partially.

The algorithm for determining the critical values ensures that each winning job pays at least the reservation price of the node it is allocated to. If there is competition from another job for the same slot, the critical value will exceed the reservation price. The question is: How can revenue R be distributed to the resource providers so as to ensure budget-balance, individual rationality and to approximate truthfulness? The VCG mechanism is generally not budget-balanced (Schnizler et al. forthcoming) and, combined with the heuristic above, not truthful (Mu'alem and Nisan 2002). A straightforward approach is to adopt the algorithm of Parkes et al. (2002) to our greedy heuristic. The basic idea of their algorithm is to approximate the VCG mechanism's truthfulness by computing discounts Δ_{parkes} that minimize the distance to the VCG-discounts Δ_{vick} while ensuring budget-balance.

In our algorithm, we first compute $\Delta_{temp,n} := V^{greedy} - (V_{-n})^{greedy}$ for each winning node n. The underlying assumption is that the greedy $\Delta_{temp,n}$ (which may in total exceed the surplus $R - \sum_{n \in N} \hat{v}_n(X^{greedy})$ from the request-side) approximate the truthful VCG-discounts $\Delta_{vick,n}$. We then solve the mathematical programme

$$[\text{BB}^{\text{greedy}}] \qquad \min_{\Delta_{greedy}} L\left(\Delta_{temp}, \Delta_{greedy}\right)$$

$$s.t. \quad \sum_{n \in N} \Delta_{greedy,n} = R - \sum_{n \in N} \hat{v}_n\left(X^{greedy}\right) \quad (C7)$$

$$0 \le \Delta_{greedy,n} \le \Delta_{temp,n}, n \in N \quad (C8)$$

The aim of $\text{BB}^{\text{greedy}}$ is to compute discounts $\Delta_{greedy,n}$ as to approximate $\Delta_{temp,n}$ in turn while ensuring (strong) budget-balance by distributing the surplus $R - \sum_{n \in N} \hat{v}_n(X^{greedy})$ to resource providers (C7).

Parkes et al. (2002) suggest and examine various distance functions analytically and numerically and recommend the "threshold function" $L_2(\Delta_{temp}, \Delta_{greedy}) := \sum_{n \in N} (\Delta_{temp,n} - \Delta_{greedys,n})^2$ as it minimizes the "residual degree of manipulation freedom" (Parkes et al. 2002), that is the maximum amount of utility a user can gain from reporting untruthfully. Analogously to the algorithm of Parkes et al. (2002), the optimal $\Delta_{greedy,n}$ can be computed without explicitly having to solve the non-linear programme $\text{BB}^{\text{greedy}}$. Let $\infty = \Delta_{temp,0} \ge \Delta_{temp,1} \ge \Delta_{temp,2} \ge \ldots \ge \Delta_{temp,|N|} \ge \Delta_{temp,|N|+1} = 0$ be the partial ordering over the temporary greedy discounts for the winning nodes. Then, for the threshold function L_2, Parkes et al. (2002) show that there is an interval K and a unique point C_t^* within this interval such that

$$\Delta_{temp,K+1} \le C_t^* = \frac{\sum_{i=1}^{K} \Delta_{temp,i} - \left(R - \sum_{n \in N} \hat{v}_n\left(X^{greedy}\right)\right)}{K} \le \Delta_{temp,K} \quad (C9)$$

and $\Delta_{greedy,n} = \max\{0, \Delta_{temp,n} - C_t^*\}$ solves $\text{BB}^{\text{greedy}}$. Node n receives a positive discount if n's temporary greedy discount is greater than the threshold C_t^*. Otherwise n does not get any discount. C_t^* can be computed by running sequentially through the partially ordered temporary discounts and checking if condition (C9) is satisfied.

Example: In the example, R = \$6,400 > \$6,040 = $\sum_{n \in N} \hat{v}_n(X^{greedy})$, i.e. there are \$360 above the reservation prices which are to be distributed as to approximate truthful payments. Solving $\text{BB}^{\text{greedy}}$ with distance function L_2 leads to C_t^* = \$1,680 and the greedy discounts listed in Table 4. The total surplus of \$360 is allocated to node N1, N2 and N3 only receive payments amounting to their reservation prices.

Node n	$\hat{v}_n(X^{greedy})$	$(V_{-n})^{greedy}$	$p_{temp,n}$	$\Delta_{temp,n}$	$p_{greedy,n}$	$\Delta_{greedy,n}$
N1	-2,700	960	-4,740	2,040	-3,060	360
N2	-1,260	2,820	-1,440	180	-1,260	0
N3	-2,080	3,000	-2,080	0	-2,080	0
Σ	**-6,040**		**-8,260**	**2,220**	**-6,400**	**360**

Table 4. Approximately truthful greedy payments.

3.4 Evaluation of the Heuristic

The greedy heuristic is designed so as to obtain approximately efficient allocations fast. It runs in polynomial time in the size of resource requests and offers: In the first phase, requests and offers get sorted in $O(|J| \cdot \log |J|)$ and $O(|N| \cdot \log |N|)$. In the second phase, the greedy allocation scheme sequentially runs through the $|J|$ sorted requests and for each job j tries to allocate j to one of the $|N|$ sorted offers. Hence the allocation phase runs in $O(|J| \cdot |N|)$. Note that the feasibility of allocating a specific job to a specific node can be tested in $O(1)$ since the number of attributes which need to be checked is constant. Inherently, this speed is only achieved at the expense of allocative efficiency compared to exact mechanisms. In a theoretical worst case analysis, the approximation of the efficient allocation can be made arbitrarily bad with respect to norm b_j , r_n suggested above by means of simple examples. The pricing scheme of the greedy mechanism consists of two parts: a truthful pricing scheme for resource requesters and an approximately truthful payment scheme for resource owners. BB^{greedy} links these two sides as to ensure budget-balance. The resulting prices are individual rational; a user reporting her true valuation will not have to pay more than her reported valuation if her request gets accepted and she will not have pay anything if she does not obtain the resources, and accordingly for resource owners.

4 RELATED WORK

The study of market mechanisms for Grid and the implementation of running Grid market prototypes have received significant attention in the past. SPAWN (Waldspurger et al. 1992) implements a market for computing resources in which each workstation auctions off idle computing time to multiple applications by means of a Vickrey auction. All resources are allocated to at most one application at a time regardless of this application's actual demand which yields low resource utilization. Chun and Culler (1999) realized a prototpyical market for computing time in which one resource provider sells computing time to multiple requesters. Resource requesters get allotted computing time proportional to their share in the total reported valuation across all requesters. The POPCORN market (Regev and Nisan 1998) is an online market for computing power which implements a Vickrey auction as well as two double auctions. All of these approaches share two major drawbacks: First, they allow the specification and trading of computing power only. But requesters require a bundle of resources such as computing power, memory, and bandwidth. On the one hand, the approaches at hand thus lead to inefficient allocations since requests with the same demand for computing power but different memory requirements, for instance, are treated the same. On the other hand, requesters are exposed to the risk of only being able to obtain one leg of the bundle of required resources without the other ("exposure risk", Schnizler et al. forthcoming). A second limitation of these approaches is that they do not support reservations of resources in advance which are essential for Quality of Service assertions.

Schnizler et al. (forthcoming) propose a comprehensive model that targets these deficiencies. They suggest the use of a multi-attribute combinatorial exchange (MACE). Users are allowed to request and offer arbitrary bundles of grid resources and can specify quality attributes on these resources. MACE implements an exact mechanism. The scheduling problem in this combinatorial setting is NP-hard, the pricing scheme of MACE yields approximately truthful prices. With truthful prices, strategic users do not have an incentive to report any valuation other than their true valuation. Due to the NP-hardness of the problem, the mechanism is adequate for batch applications – the use for interactive applications is rather limited.

The work of Bapna et al. (forthcoming) is most relevant to the work presented in this paper. In their model, multiple requesters and providers can trade both computing power and memory for a sequence of time slots. First, Bapna et al. (forthcoming) introduce an exact mechanism. By introducing fairness constraints and imposing one common valuation across all resource providers, they structure the search space in the underlying combinatorial allocation problem as to establish one common, truthful price per time slot for all accepted requests. Additionally, this introduces a linear ordering across all jobs and time which reduces the complexity of the allocation problem, which however still remains

NP-hard. To mitigate this computational complexity, they thus propose a fast, greedy heuristic at the expense of both truthfulness and efficiency.

Archer et al. (2003) also design a heuristic on the basis of Mu'alem and Nisan (2002) and Lehman et al. (2002) which, however, cannot be applied to the Grid OS context since multiple items of resources are traded, such as *x* computing cycles. The truthfulness of their mechanism, however, only holds for a small number of items.

5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In Section 1, we argued that a network-centric Grid OS coupled with market-based scheduling can increase efficiency in Grid and cluster environments by adequately allocating *all* available resources. This distinguishes network-centric Grid OS from state-of-the-art Grid middleware such as Globus, gLite, and UNICORE which rely on batch processing of *idle* resources only. While there are highly sophisticated market mechanisms available for Grid middleware (e.g. MACE), there are no mechanisms available for network-centric Grid OS which rely on interactive application processing. In Section 2 we formalized the market mechanism as a multi-attribute exchange in which resource owners and consumers can publish their demand and supply. Exact market-based scheduling mechanisms share one deficiency: they are NP-hard. While this may not be a problem in a cluster setting with a small number of users, it may prove crucial in an interactive, large-scale Grid OS environment. We thus proposed a greedy heuristic in Section 3 which performs a fast scheduling while preserving truthfulness on the request-side and approximating truthfulness on the provisioning-side of the market. In Section 4, we presented related work on market-based resource allocation in Grids.

In essence, our market mechanism suggests several intriguing research avenues. We intend to implement a prototype of the heuristic for dynamic scheduling in MOSIX, a state-of-the-art Grid OS presented in Section 1. Numerical analyses need to be performed to further analyze the heuristic's properties with respect to the economic and technical requirements in Grid OS. The approximation of efficiency needs to be compared to the optimal allocation achieved by exact mechanisms. This ratio essentially depends on the norm used in the ranking phase of the heuristic. More sophisticated norms will be developed and analyzed. The run time of the presented scheduling mechanisms needs to be evaluated to determine the critical number of requests and offers at which an exact mechanism might become infeasible and to determine the speedup achieved by the heuristic. And finally, the approximation of truthfulness on the provisioning-side of the market needs to be analyzed. It is desirable to allow users to specify dependencies between resources, such as substitutability of computing power for memory. The model in this paper hence needs to be expanded to allow for such extended bidding logics.

Acknowledgement

This work has been partially funded by EU IST programme under grant 034286 "SORMA – Self-Organizing ICT Resource Management".

References

Ali, A., A. Anjum, J. Bunn, R. Cavanaugh, F. van Lingen, R. McClatchey, M. Atif Mehmood, H. Newman, C. Steenberg, M. Thomas, I. Willers (2004). Predicting the Resource Requirements of a Job Submission. Proceedings of Computing for High Energy Physics, Interlaken, Switzerland.

Archer, A., C. Papadimitriou, K. Talwar and E. Tardos (2003). An Approximate Truthful Mechanism for Combinatorial Auctions with Single Parameter Agents. Proceedings of the 14th annual ACM-SIAM symposium on Discrete algorithms, Baltimore, Maryland, pp. 205-214.

Bapna, R., D. Sanjunkta, R. Garfinkel and J. Stallaert (forthcoming). A market design for grid computing. INFORMS Journal of Computing.

Barak, A., A. Shiloh and L. Amar (2005). An Organizational Grid of Federated MOSIX Clusters. Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid), Cardiff, Wales.

Berlich, R., M. Kunze and K. Schwarz (2005). Grid Computing in Europe: from research to deployment. Proceedings of the 2005 Australasian Workshop on Grid computing and e-research, Newcastle, Australia, 44, pp. 21-27.

Bichler, M, M. Kaukal and A. Segev (1999). Multi-attribute auctions for electronic procurement. Proceedings of the 1st IBM IAC Workshop on Internet Based Negotiation, Yorktown Heights, NY.

Buyya, R., D. Abramson and J. Giddy (2000). Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. Proceedings of the HPC ASIA'2000, the 4th International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China.

Chun, B. and D.E. Culler (1999). Market-based proportional resource sharing for clusters. Millenium Project Research Report. Available at http://www.cs.berkeley.edu/~bnc/papers/market.pdf (November 16, 2006).

Cirne, W., F. Brasileiro, N. Andrade, R. Santos, A. Andrade, R. Novaes and M. Mowbray (2006). Labs of the World, Unite! Journal of Grid Computing, 4 (3), pp. 225-246.

Ferrari, C.E. (1994). On Combinatorial Optimization Problems Arising in Computer Systems Design. Ph.D. Thesis, Technische Universität Berlin.

Foster, I., and C. Kesselman (2004). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco, 2nd edition.

Gorlatch, S., and J. Müller (2005). From Interactive Applications Toward Network-Centric Operation Systems. Workshop on Network-Centric OS, Brussels.

Lai, K., B.A. Huberman and L. Fine (2004). Tycoon: a Distributed Market-based Resource Allocation System. Technical Report, Hewlett Packard.

Lehmann, D., L.I. O'Callaghan and Y. Shoham (2002). Truth Revelation in Approximately Efficient Combinatorial Auctions. Journal of the ACM, 49 (5), pp. 577-602.

Minoli, D. (2004). A Networking Approach to Grid Computing. John Wiley & Sons, Hoboken, New Jersey.

Mu'alem, A. and N. Nisan (2002). Truthful Approximation Mechanisms for Restricted Combinatorial Auctions. In AAAI (poster). Also presented at Dagstuhl workshop on Electronic Market Design.

Myerson, R.B. and M.A. Satterthwaite (1983). Efficient mechanisms for bilateral trading. Journal of Economic Theory, 28, pp. 265-281.

Padala, P. and J.N. Wilson (2003). GridOS: Operating System Services for Grid Architectures. LNCS 2913/2003, pp. 353-362. Springer Verlag, Berlin.

Parkes, D.C., J. Kalagnanam and M. Eso (2002). Achieving budget-balance with vickrey-based payment schemes in exchanges. IBM Research Report, draft, March 13, 2002.

Regev, O. and N. Nisan (1998). The POPCORN market – an online market for computational resources. Proceedings of the 1st international conference on Information and computation economies, New York, NY, pp. 148-157.

Schnizler, B., D. Neumann, D. Veit and C. Weinhardt (forthcoming). Trading Grid Services – A Multi-attribute Combinatorial Approach. European Journal of Operational Research.

Sutherland, I.E. (1968). A futures market in computer time. Communications of the ACM, 11 (6), pp. 449-451.

Waldspurger, C., T. Hogg, B.A. Huberman, J.O. Kephart and W.S. Stornetta (1992). Spawn: A Distributed Computational Economy. IEEE Transactions on Software Engineering, 18 (2), pp. 103-117.